A wireless sensor network for tracking and localization

Michael Salib
msalib@alum.mit.edu
Dynamic Languages Group
CSAIL, MIT

BBN Technologies
April 12, 2004

Me, myself, and I

- I am finishing a combined Master's and Bachelor's degree in EECS from MIT
- I took three years off from MIT to work in industry, including consulting
- Broad experience in networks and systems
  - Focus on the practical, especially tools and protocols
  - computer architecture, fault tolerance, security, signal processing and control theory, scientific computing
- That is why my thesis is on compilers!

This talk in 30 seconds

- 10,000 Problems
- Some solutions
- Fun algorithms
  - Localization
  - Tracking
- Results
- Lessons learned
- (if we're bored) my thesis in 180 seconds

The problem

- We need money
- The answer: DARPA!
- Demo parameters
  - 200 Mica2 motes spread out over 15,000 sq ft
  - Only a few know their location
  - Deployed in a simulated urban environment
    - Lots of small buildings, hills, and other obstructions
  - Heat, rain, insects, stupid people with vehicles, in-telligant robots

Mote hardware

- 16 Mhz 8-bit Atmel CPU
- 4 KB of RAM
- 512 KB of Flash
- 2 AA batteries
- Sensors/Actuators
  - 3 colored LEDs
  - 4 Khz speaker
  - Microphone and 10-bit ADC
  - Radio with max tput of 500 bytes/sec

Mote software

- Tinyos 1.x
  - Cooperative multitasking OS with asynchronous event handlers and long running synchronous threads
    - Highly integrated with nesC, a version of C extended with new primitives to support componantized development using bidirectional interfaces
  - Safe buffer management is very tedious and error prone, but with 4K of RAM . . .
  - Buggy, broken tools, especially in the drivers

Not problems, but opportunities!

- Flash data logger is too slow to actually use
- Strange hardware interdepencies mean you cannot actually use many componants at the same time
  - Example: accurate audio sampling requires that you kill the radio since it uses the ADC
- Because of our collabarators, we could only use 1-2 radio channels
- At best, radio does 20 packets/sec, usually 10
- Packets are about 20 bytes of payload

More “opportunities”

- Simulation software did not work
- Radio reprogramming did not work
- No debugging channels
  - How do you debug a network stack?
- Motes have very fragile packages
  - Easily damaged by power cycling
  - Programming connector is only rated for 100 insertion/removal cycles
  - Connectors are difficult to manipulate, especially after making a code change to the 199th mote

Localization

- Local measurements are easy
  - All nodes have GUID
  - Use thunder/lightning protocol to determine range to nearby neighbors
- Use gradient propogation combined with range estimates to form a local coordinate system based on the gradient anchors
- But we want to impose a global coordinate system!

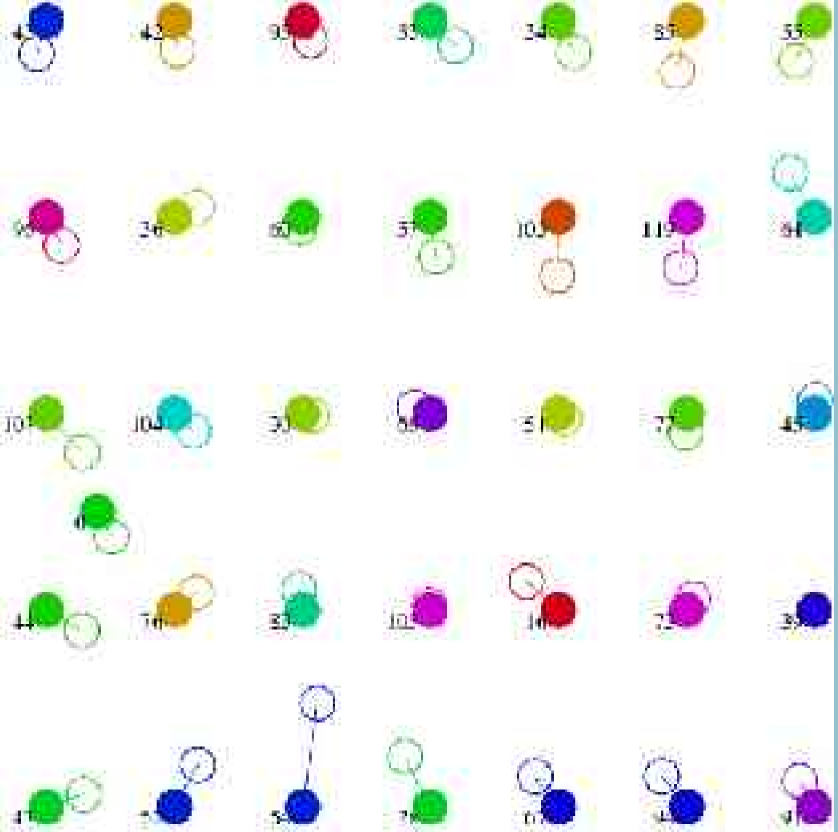
Think locally, act globally

- Every node knows its distance to each anchor
- Node position is chosen to minimize the difference between the node's estimate of its distance to each anchor and the measured distance
- Minimization is performed iteratively on each node using gradient descent
- Accuracy is improved by computing straightness factors between pairs of anchors and using them to compensate measurements

Localization “opportunities”

- Accoustic ranging does not work indoors
- Finding a place to deploy 20-30 motes outside, in Cambridge, with AC power nearby is difficult
- Testing is very labor intensive. Mostly my labor.
- Do most computations on the host PC just to get something that can be debugged

Localization results



- Accoustic ranging is very accurate, when it works
- Localization error is about 10% in dense networks

Tracking algorithm

- Objects being tracked carry “tags” that are really motes
- Tags broadcast their ID and the current time
- Nodes that hear a tag inform the base station of their ID, the tag ID, and the tag time
- Multihop transport uses gradient routing
- Gradient routing
  - Directed flooding, “up” the gradient to the dest
  - Culls duplicates and stale reports: info now is much more important than info then
  - Aggregates messages

Tracking "opportunities"

- Net throughput absolutely dominates
- Batching, dup elimination, and culling stale results are huge wins
- Smarter systems are obvious
  - Do more "local" computation and only send one result with the exact position to the host
  - But how do you debug that code?
  - We are utterly at the mercy of our pathetic tools
- Its always the little things
  - It turns out that the Java packet multiplexing code uses UDP and not TCP. Oops.

Tracking results

- It basically worked (with hand-holding)
- Tracking latency is about 1-2 seconds

Lessons learned

- Testing time does not count when the Col won't let you actually test anything
- Doing things the quick and easy way will bite you
  - I've learned this one many times before, so why does management continue to teach it to me?
- No project is so simple that it cannot be de-railed by rotten tools and understaffing
- People do not work any better when being shot at, they just work more frantically